PROGRAMACIÓN PARA SISTEMAS

TEMA 4

Unix: Shell y Scripts

Unix: Shell y Scripts

INTRODUCCIÓN

- LOS PUNTOS QUE SE TRATARÁN SON LOS SIGUIENTES:
  - Comando read para lectura desde teclado o fichero
  - Comando printf para salidas con formato
  - Recomendaciones para escribir scripts
  - Tratamiento de las opciones mediante el comando *getopts*

BIBLIOGRAFÍA RECOMENDADA:

- ***The UNIX Programming Environment, B. Kernighan, R. Pike***
- ***BASH Cookbook, C. Albing, JP. Vossen, C. Newham***
- ***http://www.gnu.org/software/bash/manual/bashref.html***

Unix: Shell y Scripts

COMANDO DE LECTURA *read*

- LEE UNA LÍNEA DESDE LA ENTRADA ESTÁNDAR O DESDE UN FICHERO ESPECIFICADO MEDIANTE UN DESCRIPTOR.
- LECTURA DESDE ENTRADA ESTÁNDAR (CASO TÍPICO):

  read [-d delimitador] [-n nchar] [–p prompt] variables
- EJEMPLO 1: La primera palabra va a la primera variable, el resto de la línea va a la segunda variable

```
read A B; echo $A; echo $B
Hola Holita Hola  ← Entrada desde teclado
Hola ← Variable A
Holita Hola ← Variable B
```

Unix: Shell y Scripts

COMANDO DE LECTURA *read*

- EJEMPLO 2 : Si no se especifican variables, la línea pasa a la variable REPLAY:

```
read ; echo $REPLY
                Hola Holita Hola  ← Entrada desde teclado
                Hola  Holita Hola← Variable REPLY
```

- EJEMPLO 3 : La opción *–n nchar* lee sólo n caracteres de la línea sin esperar al *newline*:

```
read –n 4; echo “”; echo $REPLY
                Hola  ← Entrada desde teclado (no admite más)
                Hola  ← Variable REPLY
```

Unix: Shell y Scripts

COMANDO DE LECTURA *read*

- EJEMPLO 4 : El delimitador por defecto de la línea es *newline*. Se puede declarar otro (un solo carácter) mediante la opción *–d delimitador:*

```
read –d  : ; echo $'\n'$REPLY ← El delimitador es ahora ":"
                Hola ← Entrada desde teclado
                Holita              :
                Hola :              :
                Hola  Holita Hola← Variable REPLY
```

- EJEMPLO 5 : Por defecto no hay *prompt* de lectura. Se puede declarar uno mediante la opción *–p prompt*

```
read –p “Teclee su respuesta (s/n)>”;  echo $REPLY
                Teclee su respuesta (s/n)>s  ← Entrada desde teclado
                s ← Variable REPLY
```

Unix: Shell y Scripts

COMANDO DE LECTURA *read*

- LECTURA DE UN VECTOR DESDE ENTRADA ESTÁNDAR :

  read [-d delimitador] [-n nchar] [–p prompt] -a vector
- EL VECTOR SE CREA AUTOMÁTICAMENTE, SIN NECESIDAD DE EMPLEAR EL COMANDO *declare –a vector,* Y SE ASIGNA A SUS COMPONENTES VALORES DE LA LÍNEA SEPARADOS POR EL SEPARADOR DE CAMPO (POR DEFECTO ESPACIO)
- EJEMPLO 6: Separador por defecto

```
read –a vector; echo ${vector[0]}; echo ${vector[1]}
                A B  ← Entrada desde teclado
                A  ← Variable vector[0]
                B  ← Variable vector[1]
```

Unix: Shell y Scripts

COMANDO DE LECTURA *read*

- EJEMPLO 7: Con separador ","

IFS=',' read –a vector; echo ${vector[0]}; echo ${vector[1]}
A, B ← Entrada desde teclado
A ← Variable vector[0]
B ← Variable vector[1]

- EJEMPLO 8: Con delimitador

read –a vector –d : ; echo ""; echo ${vector[0]}; echo ${vector[1]}
A ← Entrada desde teclado
B: ← Entrada desde teclado
A ← Variable vector[0]
B ← Variable vector[1]

Unix: Shell y Scripts

COMANDO DE LECTURA *read*

- LECTURA DESDE FICHERO ASCII

read –u descriptor_fichero vector variables

- EL DESCRIPTOR SE ASIGNA MEDIANTE EL COMANDO

exec n<fichero (Equivalente a open)

- EL COMANDO read ASIGNA VALORES A LAS VARIABLES DE ACUERDO AL CONTENIDO DE LA LÍNEA Y AL SEPARADOR DE CAMPOS
- LA REPETICIÓN DEL COMANDO DA LUGAR A SUCESIVAS LECTURAS DE LÍNEAS HASTA DETECTAR EOF, LO QUE SE INDICA CON UN STATUS 1
- PARA CERRAR EL FICHERO Y LIBERAR EL DESCRIPTOR:

exec n>&- (Equivalente a close)

Unix: Shell y Scripts

COMANDO DE LECTURA *read*

- EJEMPLO 9 :

Fichero file: *uno dos*
tres cuatro

exec 5<file; ← Asignar descriptor de fichero
read –u 5 A B ; echo "$A $B" ← Primera lectura
uno dos ← Variables A y B
read –u 5 C D ; echo "$C $D" ← Segunda lectura
tres cuatro ← Variables C y D
read –u 5; echo $? ← Detecta EOF
1 ← Status de comando read
exec 5>&- ← Libera descriptor

Unix: Shell y Scripts

COMANDO DE ESCRITURA CON FORMATO *printf*

- ESCRIBE EL CONTENIDO DE LOS ARGUMENTOS EN LA SALIDA ESTÁNDAR O EN OTRA VARIABLE DE ACUERDO CON UN FORMATO:

printf [-v variable] formato [argumentos]

- EL FORMATO ES SIMILAR AL EMPLEADO EN EL LENGUAJE C
- EJEMPLO 10:

Text="Texto"
Int=7
Flot=3.14159265358979
printf "Text=%s Int=%5d Flot=%6.4f " $Text $Int $Flot

Resultado:
Text=Texto Int= 7 Flot=3.1416

Unix: Shell y Scripts

COMANDO DE ESCRITURA CON FORMATO *printf*

- EJEMPLO 11: Manejo de errores

```
E_BADDIR=85
var=directorio_no_existente
error()
{
        printf "$@" >&2 # Salida de mensaje a stderr
        echo exit $E_BADDIR
}
cd $var || error $"No existe el directorio %s." "$var"
```

Unix: Shell y Scripts

COMANDO DE ESCRITURA CON FORMATO *printf*

- EJEMPLO 12: Salida a variable

printf -v Variable "%s %5d %6.4f " $Text $Int $Flot
echo $Variable
Text=Texto Int= 7 Flot=3.1416 ← Contenido de Variable

- ¡LOS VALORES GUARDADOS EN LA VARIABLE ANTERIOR PUEDEN RECUPERARSE COMO PARÁMETROS POSICIONALES MEDIANTE EL COMANDO *set* ! :

set $Variable; echo $1; echo $2; echo $3
Texto ← Contenido de parámetro posicional 1
7 ← Contenido de parámetro posicional 2
3.1416 ← Contenido de parámetro posicional 3

Unix: Shell y Scripts

RECOMENDACIONES PARA LOS SCRIPTS

- EL SCRIPT DEBE ALMACENARSE EN EL DIRECTORIO $HOME/bin Y TENER PERMISO DE EJECUCIÓN
- LA PRIMERA LÍNEA DE UN SCRIPT DE BASH ES EL *shebang* :

  #!/bin/bash

  NO ES UN COMENTARIO, SINO UNA INDICACIÓN DE QUE EL SCRIPT CORRERÁ EN EL SHELL *bash*.
- EN LA TERMINACIÓN SE DEBE EMPLEAR EL COMANDO *exit* SEGUIDO DE UN NÚMERO DE STATUS (SALIDA NORMAL=0, ERROR>0)
- SE DEBEN EMPLEAR COMENTARIOS (PRECEDIDOS POR #) E INDENTACIÓN (COMO EN C) PARA FACILITAR LA LECTURA Y COMPRENSIÓN DEL CÓDIGO

Unix: Shell y Scripts

RECOMENDACIONES PARA LOS SCRIPTS

- AL MENOS DEBERÍA EXISTIR LA OPCIÓN *–h* PARA MOSTRAR UNA AYUDA SOBRE EL SCRIPT
- EN SU DEFECTO O SIMULTÁNEAMENTE, SI ES POSIBLE, DEBE EXISTIR UNA SALIDA DE USO (*usage*) CUANDO SE TECLEA ÚNICAMENTE EL NOMBRE DEL SCRIPT SIN ARGUMENTOS
- LOS MENSAJES DE ERROR DEBEN MOSTRARSE EN LA SALIDA ESTÁNDAR DE ERROR CON EL SIGUIENTE FORMATO

  Nombre_script: ERROR (Código error), Descripción_breve_del_error
  Nombre_script: Descripción_detallada_del_error

Unix: Shell y Scripts

EL MANEJO DE OPCIONES MEDIANTE *getopts*

- EL COMANDO *getopts* PERMITE INTERPRETAR LAS OPCIONES DE UN SCRIPT DE FORMA SENCILLA
- DESGRACIADAMENTE SÓLO TRATA LAS OPCIONES EN FORMATO CORTO DE LA FORMA:

  nombre_script –a argumentos

  nombre_script –a –b argumentos

  nombre_script –ab argumentos

  nombre_script –a –b valorb argumentos

  DONDE *a, b, etc* SON OPCIONES CON O SIN VALOR.
- NO TRATA OPCIONES CON FORMATO LARGO (*--opción*)

Unix: Shell y Scripts

EL MANEJO DE OPCIONES MEDIANTE *getopts*

- EJEMPLO 13:

```
#! /bin/bash
aflag=
bflag=
while getopts ':ab:' OPTION ← (:) Silenciar getopts, (ab) Opciones, (:) arg tras b
do
        case $OPTION in
        a)      aflag=1 ;;
        b)      bflag=1
                bval="$OPTARG" ;;
        \:)     printf "Falta valor en -%s\n" $OPTARG
                exit 2 ;;
        \?)     printf "Opcion desconocida -%s\n" $OPTARG
                exit 2 ;;
esac >&2
shift $(($OPTIND – 1))
```